Dhruv Singal
349-01
1.

1.a) Worst case complexity of Selection Sort.
problem size = n ← # elements in data[].
basic operation = key comparison
worst case = # of basic operations doesn't depend on data[] layout.

total # comparisons = ?
count comparisons inside out.

$i=0,\ j=1 \ldots\ldots n-1 \Rightarrow n-1$ key comparisons.
$i=1,\ j=2 \ldots\ldots n-1 \rightarrow n-2$ " "
$i=2,\ j=3 \ldots\ldots n-1 \rightarrow n-3$ " "
⋮
$i=n-2,\ j=n-1 \ldots\ldots n-1 \rightarrow 1$ " "

$\therefore$ # total comparisons $= 1+2+3+4+\ldots\ldots n-1$
$= \sum_{i=1}^{n-1} \Rightarrow T(n) = ?$

sum of AP: $\frac{n}{2}(2a+(n-1)d) = \frac{(n-1)}{2}(2+(n-1-1)1)$ → $a=1,\ n=n-1,\ d=1.$

$= \frac{n(n-1)}{2} = \theta(n^2).$

1.b) Worst case complexity of insertion sort.
problem size n ← # of elements in data[]
basic operation = key comparison
worst case = the array data[] is presorted in the reverse order
basic operations inside out:
n-1 times through outer loop
summation based on i.

comparisons: inside out:
$i=1$ # comparisons: 1
$i=2$ # " $= 2$.

i goes from 1 to n-1.

$\therefore$ total # comparisons $= 1+2+3+ \cdots\cdots n-1$

$= \frac{n(n-1)}{2}$ comparisons.

$= O(n^2)$ in the worst case

1.c: worst case complexity of merge sort

problem size: # items in data[] array

basic operation: key comparison.

worst case: data entries are interleaved ascending, n is a power of two.

total number of comparisons = ?

M(n) = # comparisons to mergesort n-items in worst case.

$$M(n) = M(\tfrac{n}{2}) + M(\tfrac{n}{2}) + n = 2M(\tfrac{n}{2}) + n$$

$$= 2(2M(\tfrac{n}{4}) + \tfrac{n}{2}) + n = 4M(\tfrac{n}{4}) + 2n.$$

$$= 8M(\tfrac{n}{8}) + 3n$$

$$\vdots$$

$$= 2^n M(1) + n\log_2 n = n\log_2 n \quad \because M(1) = 0$$

$\therefore$ # total comparisons $= n\log_2 n$

$\therefore$ time complexity in worst case $= O(n\log n)$

2.

Table 1

Number of Elements	Selection Sort		
	sorted	revsorted	random
10	45	45	45
1000	499500	499500	499500
2000	1999000	1999000	1999000
3000	4498500	4498500	4498500
4000	7998000	7998000	7998000

Table 2

Number of Elements	Insertion Sort		
	sorted	revsorted	random
10	9	45	21
1000	999	499450	251582
2000	1999	1998795	1016590
3000	2999	4498052	2269084
4000	3999	7997251	3958487

Table 3

Number of Elements	Merge Sort		
	sorted	revsorted	random
10	5	5	9
1000	500	500	995
2000	1003	1000	1998
3000	1500	1500	2998
4000	2000	2000	3999

3. The numbers in the table agree with the theoretical analysis. I wasn't able to verify the worst case scenario for the merged sort but looking at multiple runs of merge sort on random data, the maximum number of comparisons I saw approached the theoretical limit of nlog2n. For insertion and selection sort, the worst case total number of key comparisons exactly matched the theoretical analysis.